Identifying Clusters using Node2Vec, Spectral Clustering and GCN

Aayush Shrivastava Vivek Tarachandani Ankit Pandey Anish Pawar Puneet Kumar Rajan
{cs24mtech02001, ai23mtech14008, ai23mtech13002, ai23mtech14002, ai22mtech11010}@iith.ac.in

Indian Institute of Technology, Hyderabad

Abstract

Node2Vec is a framework for learning continuous feature representations for nodes in networks. Spectral Clustering is a technique that uses eigenvalues of the similarity matrix of the data to reduce dimensions before clustering in fewer dimensions. This method is particularly effective for identifying clusters that are not necessarily globular and can capture complex cluster structures that traditional clustering methods might fail to detect. Graph Convolutional Networks (GCN) are powerful neural network architectures for processing data from graphs.

1. Problem Statement

The Node2Vec project aims to develop an algorithm that generates robust feature representations for network nodes using biased random walks, enhancing tasks like node classification and community detection while preserving network topology. The Spectral Clustering project aims to partition graphs into clusters based on the eigenvalues of their similarity matrices, enabling the identification of complex community structures that traditional clustering methods might miss. The GCN project seeks to develop a neural network architecture that leverages node features and graph structure to effectively perform tasks like node classification, link prediction, and graph classification within graph-based data. This approach aims to enhance the learning of node representations by incorporating neighborhood information directly into the learning process, enabling more accurate predictions and insights into the underlying graph structure.

2. Dataset Description

The dataset "Payments.csv" contains 1,30,535 rows and three columns: Sender (Integer), Receiver (Integer), and Amount (Integer). Each row represents a transaction where a certain amount is transferred from the Sender to the Receiver. Out of these transactions, only 5,358 involve a unique pair of dealers interacting with each other. These 5,358 transactions are unique interactions between pairs of dealers who have conducted business with each other at least once.

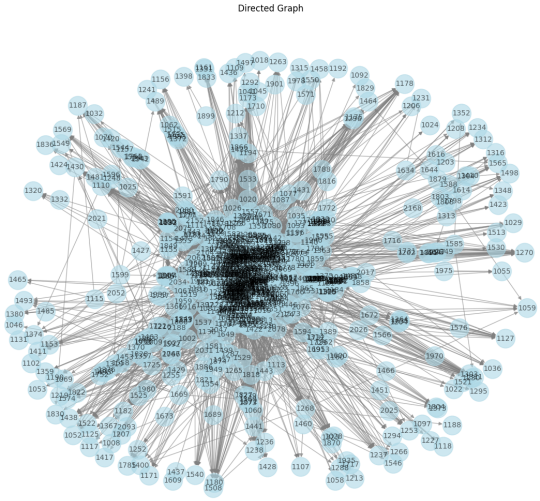

Figure 1: Graphical Representation of Payments.csv

The graphical representation of the given dataset comprises 799 nodes and 5,358 edges. Notably, not all nodes are interconnected. Some nodes lack outgoing edges, rendering them disconnected from other nodes.

Upon data analysis, the highest transaction recorded is 2,124,500, while the lowest transaction is 1501.

3. Data Exploration

The assignment includes one file - Payments.csv. The dataset contains transaction data between the Sender and Receiver, with columns representing the Sender ID, Receiver ID, and the transaction value as Amount. A graph is created using the Sender and Receiver IDs, with Amount as edge weights, as shown in Figure 1.

There are 1,30,535 entries in the Payments.csv, which

represents the transaction between the sender and the receiver and the value of the transaction amount.

4. Node2Vec

Node2Vec framework is based on learning continuous feature representation for nodes in the graph and preserving the knowledge gained from the neighboring 100 nodes. Let us understand how the algorithm works. Say we have a graph with a few interconnected nodes creating a network. So, the Node2Vec algorithm learns a dense representation (say 100 dimensions/features) of every node in the network. The algorithm suggests that if we plot these 100 dimensions of each node in a 2-dimensional graph by applying PCA, the distance of the two nodes in that low-dimensional graph would be the same as their actual distance in the given network.

Algorithm 1: Node2VecWalk

Data: Graph $G' = (V, E, \pi)$, Start Node u, Length l
Result: A walk of length l starting from node u

Initialize walk to $[u]$
for *walk_iter* $= 1$ *to* l **do**
  $curr \leftarrow$ walk$[-1]$
  $V_{curr} \leftarrow$ GetNeighbors$(curr, G')$
  $s \leftarrow$ AliasSample(V_{curr}, π)
  Append s to walk
return *walk*

Algorithm 2: Learn_Features

Data: Graph $G = (V, E, W)$
Result: Feature representation f for all nodes in V

Parameters: Dimensions d, Walks per node r, Walk length l, Context size k, Return p, In-out q;
$\pi \leftarrow$ PreprocessModifiedWeights(G, p, q);
$G' \leftarrow (V, E, \pi)$;
Initialize walks $\leftarrow \emptyset$;
for *iter* $\leftarrow 1$ **to** r **do**
  forall $u \in V$ **do**
    walk $\leftarrow$ Node2VecWalk(G', u, l);
    Append walk to walks;
  end
end
$f \leftarrow$ StochasticGradientDescent$(k, d,$ walks$)$;
return f;

5. Spectral Clustering

Spectral Clustering is a variant of the clustering algorithm that uses the connectivity between the data points to form the clustering. It uses eigenvalues and eigenvectors of the data matrix to forecast the data into lower dimensions space to cluster the data points. It is based on a graph representation of data where the data points are represented as nodes and an edge represents their similarity.

Algorithm 3: SpectralClustering(D, s, k)

Input: D: a set of data points $\{x_1, \ldots, x_n\}$, $s(x, x')$: a similarity function, k: number of clusters to construct
Output: Clusters $A_1, \ldots, A_k$

Construct a similarity graph G from D using s
Let W be the weighted adjacency matrix of G
Compute the graph Laplacian L
Compute the first k eigenvectors $u_1, \ldots, u_k$ of L
Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing $u_1, \ldots, u_k$ as columns
for $i = 1, \ldots, n$ **do**
  Let $z_i \in \mathbb{R}^k$ be the i^{th} row of U
end
Cluster the points z_i with the k-means algorithm into clusters $C_1, \ldots, C_k$
for $i = 1, \ldots, k$ **do**
  Let $A_i = \{x_j \mid z_j \in C_i\}$
end
return $A_1, \ldots, A_k$

6. Graph Convolutional Network (GCN)

Graph Convolutional Networks (GCNs) generalize convolutional neural networks to graph-structured data. They operate by learning a function that takes a graph's adjacency matrix, a degree matrix, and a feature matrix as input and produces node-level embeddings as output. These embeddings can then be used in various downstream tasks such as node classification, link prediction, or graph classification.

Algorithm 4: GraphConvolutionalNetwork (GCN)

Input: Graph $G = (V, E)$ with adjacency matrix A, degree matrix D, feature matrix X, depth K, non-linearity σ, learnable weight matrices $W^{(k)}$, $k = 1, \ldots, K$
Output: Node-level embeddings Z

$H^{(0)} \leftarrow X$
for $k = 1, \ldots, K$ **do**
  $H^{(k)} \leftarrow \sigma\left(D^{-\frac{1}{2}} A D^{-\frac{1}{2}} H^{(k-1)} W^{(k)}\right)$
end
$Z \leftarrow H^{(K)}$
return Z

7. Results

We obtain dense vector embeddings for each node upon applying the Node2Vec algorithm to our graph data. These embeddings capture the local neighborhood structure and encode the nodes' positions within the overall graph topology. Leveraging these embeddings, we utilize the k-means clustering algorithm to partition nodes into distinct clusters, ideally corresponding to different communities or roles within the graph.

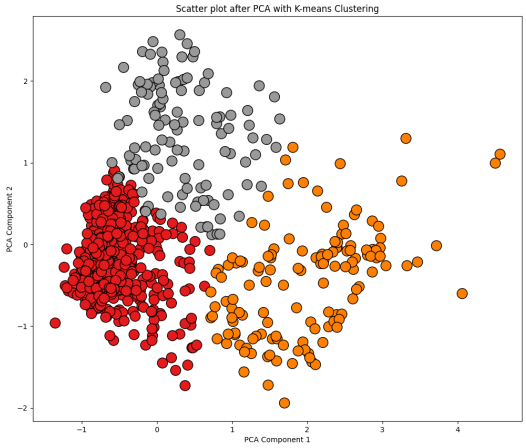

Figure 2: Node2Vec Output

In contrast, when employing Spectral Clustering, the approach capitalizes on the eigenvalues of the graph's Laplacian matrix to perform dimensionality reduction before clustering. This method is particularly adept at identifying non-linearly separable clusters in the original space, often revealing global structures and groupings that are not immediately apparent.

Graph Convolutional Networks (GCN), on the other hand, provide a more nuanced method of embedding generation. By iteratively aggregating and transforming neighbor information, GCNs produce embeddings that inherently incorporate both local graph topology and node features. These embeddings can then be clustered like those from Node2Vec to discover community structures.

8. Conclusion

In conclusion, the assignment employing Node2Vec, Spectral Clustering, and GCN techniques demonstrates the versatility and effectiveness of various graph-based approaches in uncovering latent structures within complex networks. Each method offers unique advantages: Node2Vec captures local neighborhood information, Spectral Clustering reveals global structures, and GCNs integrate node features and graph topology. The choice among these techniques depends on the specific characteristics of the data and the objectives of the analysis, showcasing the importance of tailored algorithm selection for optimal results in graph analytics tasks. According to the results, GCN provides the most accurate clustering, followed by Node2Vec and Spectral Clustering.

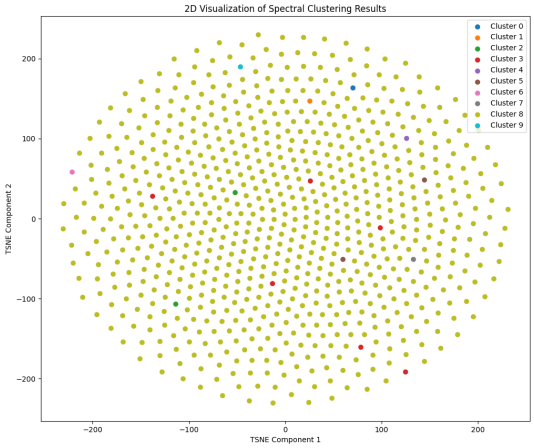

Figure 3: Spectral Clustering Output

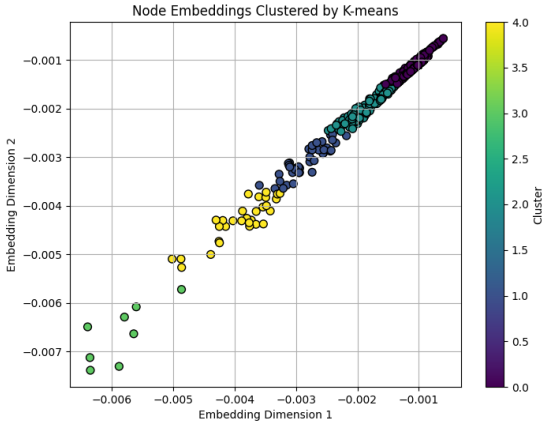

Figure 4: GCN Output

Node2Vec Code is available in the given link.
Spectral Clustering Code is available in the given link.
GCN Code is available in the given link.

References

[1] Node2Vec: Scalable Feature Learning for Networks (2016), Aditya Grover, Jure Leskovec.

[2] A Tutorial on Spectral Clustering (2007), Ulrike von Luxburg.

[3] Semi-Supervised Classification with Graph Convolutional Networks (2016), Thomas N. Kipf.